CHAPTER 15. MAIL SERVERS

Red Hat Enterprise Linux offers many advanced applications to serve and access email. This chapter describes modern email protocols in use today, and some of the programs designed to send and receive email.

15.1. EMAIL PROTOCOLS

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

15.1.1. Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the Simple Mail Transfer Protocol (SMTP).

15.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. But more intermediate SMTP servers may be included in this chain. This concept is called a mail relaying. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or spam possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called open relay servers.

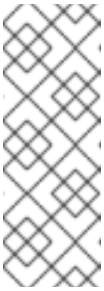
Red Hat Enterprise Linux 7 provides the Postfix and Sendmail SMTP programs.

15.1.2. Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP).

15.1.2.1. POP

*The default POP server under Red Hat Enterprise Linux is **Dovecot** and is provided by the **dovecot** package.*



NOTE

To install ***Dovecot*** *run the following command:*

```
~]# yum install dovecot
```

For more information on installing packages with Yum, see Section 9.2.4, "Installing Packages".

When using a ***POP*** *server, email messages are downloaded by email client applications. By default, most* ***POP*** *email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.*

POP *is fully compatible with important Internet messaging standards, such as Multipurpose Internet Mail Extensions (MIME), which allow for email attachments.*

POP *works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections,* ***POP*** *requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.*

The most current version of the standard ***POP*** *protocol is* ***POP3****.*

There are, however, a variety of lesser-used ***POP*** *protocol variants:*

- *APOP – **POP3** with **MD5** authentication. An encoded hash of the user's password is sent from the email client to the server rather than sending an unencrypted password.*
- *KPOP – **POP3** with Kerberos authentication.*
- *RPOP – **POP3** with **RPOP** authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so **RPOP** is no more secure than standard **POP**.*

To improve security, you can use Secure Socket Layer (SSL) encryption for client authentication and data transfer sessions. To enable SSL encryption, use:

- *The **pop3s** service*
- *The **stunnel** application*
- *The **starttls** command*

For more information on securing email communication, see Section 15.5.1, "Securing Communication".

15.1.2.2. IMAP

The default ***IMAP*** *server under Red Hat Enterprise Linux is* ***Dovecot*** *and is provided by the* ***dovecot*** *package. See Section 15.1.2.1, "POP" for information on how to install* ***Dovecot****.*

When using an ***IMAP*** *mail server, email messages remain on the server where users can read or delete them.* ***IMAP*** *also allows client applications to create, rename, or delete mail directories on the server to organize and store email.*

IMAP *is particularly useful for users who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header*

information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

*For convenience, **IMAP** client applications are capable of caching copies of messages locally, so the user can browse previously read messages when not directly connected to the **IMAP** server.*

***IMAP**, like **POP**, is fully compatible with important Internet messaging standards, such as MIME, which allow for email attachments.*

*For added security, it is possible to use **SSL** encryption for client authentication and data transfer sessions. This can be enabled by using the **imaps** service, or by using the **stunnel** program.*

- *The **pop3s** service*
- *The **stunnel** application*
- *The **starttls** command*

For more information on securing email communication, see Section 15.5.1, "Securing Communication".

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality.

15.1.2.3. Dovecot

*The **imap-login** and **pop3-login** processes which implement the **IMAP** and **POP3** protocols are spawned by the master **dovecot** daemon included in the **dovecot** package. The use of **IMAP** and **POP** is configured through the **/etc/dovecot/dovecot.conf** configuration file; by default **dovecot** runs **IMAP** and **POP3** together with their secure versions using **SSL**. To configure **dovecot** to use **POP**, complete the following steps:*

1. *Edit the **/etc/dovecot/dovecot.conf** configuration file to make sure the **protocols** variable is uncommented (remove the hash sign (**#**) at the beginning of the line) and contains the **pop3** argument. For example:*

   ```
   protocols = imap pop3 lmtp
   ```

   *When the **protocols** variable is left commented out, **dovecot** will use the default values as described above.*

2. *Make the change operational for the current session by running the following command as **root**:*

   ```
   ~]# systemctl restart dovecot
   ```

3. *Make the change operational after the next reboot by running the command:*

   ```
   ~]# systemctl enable dovecot
   Created symlink from /etc/systemd/system/multi-user.target.wants/dovecot.service to
   /usr/lib/systemd/system/dovecot.service.
   ```

NOTE

*Please note that **dovecot** only reports that it started the **IMAP** server, but also starts the **POP3** server.*

*Unlike **SMTP**, both **IMAP** and **POP3** require connecting clients to authenticate using a user name and password. By default, passwords for both protocols are passed over the network unencrypted.*

*To configure **SSL** on **dovecot**:*

- *Edit the **/etc/dovecot/conf.d/10-ssl.conf** configuration to make sure the **ssl_protocols** variable is uncommented and contains the **!SSLv2 !SSLv3** arguments:*

```
ssl_protocols = !SSLv2 !SSLv3
```

  *These values ensure that **dovecot** avoids SSL versions 2 and also 3, which are both known to be insecure. This is due to the vulnerability described in POODLE: SSLv3 vulnerability (CVE-2014-3566). See Resolution for POODLE SSL 3.0 vulnerability (CVE-2014-3566) in Postfix and Dovecot for details.*

  *Make sure that **/etc/dovecot/conf.d/10-ssl.conf** contains the following option:*

```
ssl=required
```

- *Edit the **/etc/pki/dovecot/dovecot-openssl.cnf** configuration file as you prefer. However, in a typical installation, this file does not require modification.*
- *Rename, move or delete the files **/etc/pki/dovecot/certs/dovecot.pem** and **/etc/pki/dovecot/private/dovecot.pem**.*
- *Execute the **/usr/libexec/dovecot/mkcert.sh** script which creates the **dovecot** self signed certificates. These certificates are copied in the **/etc/pki/dovecot/certs** and **/etc/pki/dovecot/private** directories. To implement the changes, restart **dovecot** by issuing the following command as **root**:*

```
~]# systemctl restart dovecot
```

*More details on **dovecot** can be found online at http://www.dovecot.org.*

15.2. EMAIL PROGRAM CLASSIFICATIONS

In general, all email applications fall into at least one of three classifications. Each classification plays a specific role in the process of moving and managing email messages. While most users are only aware of the specific email program they use to receive and send messages, each one is important for ensuring that email arrives at the correct destination.

15.2.1. Mail Transport Agent

*A Mail Transport Agent (MTA) transports email messages between hosts using **SMTP**. A message may involve several MTAs as it moves to its intended destination.*

While the delivery of messages between machines may seem rather straightforward, the entire process of deciding if a particular MTA can or should accept a message for delivery is quite complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's configuration or the access configuration for the network on which the MTA resides.

Some email client programs, can act as an MTA when sending an email. However, such email client programs do not have the role of a true MTA, because they can only send outbound messages to an MTA they are authorized to use, but they cannot directly deliver the message to the intended recipient's email

server. This functionality is useful if host running the application does not have its own MTA.

Since Red Hat Enterprise Linux offers two MTAs, Postfix and Sendmail, email client programs are often not required to act as an MTA. Red Hat Enterprise Linux also includes a special purpose MTA called Fetchmail.

For more information on Postfix, Sendmail, and Fetchmail, see Section 15.3, "Mail Transport Agents".

15.2.2. Mail Delivery Agent

A Mail Delivery Agent (MDA) is invoked by the MTA to file incoming email in the proper user's mailbox. In many cases, the MDA is actually a Local Delivery Agent (LDA), such as ***mail*** *or Procmail.*

Any program that actually handles a message for delivery to the point where it can be read by an email client application can be considered an MDA. For this reason, some MTAs (such as Sendmail and Postfix) can fill the role of an MDA when they append new email messages to a local user's mail spool file. In general, MDAs do not transport messages between systems nor do they provide a user interface; MDAs distribute and sort messages on the local machine for an email client application to access.

15.2.3. Mail User Agent

A Mail User Agent (MUA) is synonymous with an email client application. MUA is a program that, at a minimum, allows a user to read and compose email messages. MUAs can handle these tasks:

- *Retrieving messages via the* ***POP*** *or* ***IMAP*** *protocols*
- *Setting up mailboxes to store messages.*
- *Sending outbound messages to an MTA.*

MUAs may be graphical, such as ***Thunderbird***, ***Evolution***, *or have simple text-based interfaces, such as* ***mail*** *or* ***Mutt***.

15.3. MAIL TRANSPORT AGENTS

Red Hat Enterprise Linux 7 offers two primary MTAs: Postfix and Sendmail. Postfix is configured as the default MTA and Sendmail is considered deprecated. If required to switch the default MTA to Sendmail, you can either uninstall Postfix or use the following command as ***root*** *to switch to Sendmail:*

```
~]# alternatives --config mta
```

You can also use the following command to enable the desired service:

```
~]# systemctl enable service
```

Similarly, to disable the service, type the following at a shell prompt:

```
~]# systemctl disable service
```

For more information on how to manage system services in Red Hat Enterprise Linux 7, see Chapter 10, Managing Services with systemd .

15.3.1. Postfix

Originally developed at IBM by security expert and programmer Wietse Venema, Postfix is a Sendmail-compatible MTA that is designed to be secure, fast, and easy to configure.

To improve security, Postfix uses a modular design, where small processes with limited privileges are launched by a master daemon. The smaller, less privileged processes perform very specific tasks related to the various stages of mail delivery and run in a changed root environment to limit the effects of attacks.

Configuring Postfix to accept network connections from hosts other than the local computer takes only a few minor changes in its configuration file. Yet for those with more complex needs, Postfix provides a variety of configuration options, as well as third party add-ons that make it a very versatile and full-featured MTA.

The configuration files for Postfix are human readable and support upward of 250 directives. Unlike Sendmail, no macro processing is required for changes to take effect and the majority of the most commonly used options are described in the heavily commented files.

15.3.1.1. The Default Postfix Installation

*The Postfix executable is **postfix**. This daemon launches all related processes needed to handle mail delivery.*

*Postfix stores its configuration files in the **/etc/postfix/** directory. The following is a list of the more commonly used files:*

- ***access** – Used for access control, this file specifies which hosts are allowed to connect to Postfix.*
- ***main.cf** – The global Postfix configuration file. The majority of configuration options are specified in this file.*
- ***master.cf** – Specifies how Postfix interacts with various processes to accomplish mail delivery.*
- ***transport** – Maps email addresses to relay hosts.*

*The **aliases** file can be found in the **/etc** directory. This file is shared between Postfix and Sendmail. It is a configurable list required by the mail protocol that describes user ID aliases.*

IMPORTANT

*The default **/etc/postfix/main.cf** file does not allow Postfix to accept network connections from a host other than the local computer. For instructions on configuring Postfix as a server for other clients, see Section 15.3.1.3, "Basic Postfix Configuration".*

*Restart the **postfix** service after changing any options in the configuration files under the **/etc/postfix/** directory in order for those changes to take effect. To do so, run the following command as **root**:*

```
~]# systemctl restart postfix
```

15.3.1.2. Upgrading From a Previous Release

The following settings in Red Hat Enterprise Linux 7 are different to previous releases:

- ***disable_vrfy_command = no** – This is disabled by default, which is different to the default for Sendmail. If changed to **yes** it can prevent certain email address harvesting methods.*

- ***allow_percent_hack = yes*** – *This is enabled by default. It allows removing* ***%*** *characters in email addresses. The percent hack is an old workaround that allowed sender-controlled routing of email messages.* ***DNS*** *and mail routing are now much more reliable, but Postfix continues to support the hack. To turn off percent rewriting, set* ***allow_percent_hack*** *to* ***no****.*

- ***smtpd_helo_required = no*** – *This is disabled by default, as it is in Sendmail, because it can prevent some applications from sending mail. It can be changed to* ***yes*** *to require clients to send the HELO or EHLO commands before attempting to send the MAIL, FROM, or ETRN commands.*

15.3.1.3. Basic Postfix Configuration

By default, Postfix does not accept network connections from any host other than the local host. Perform the following steps as ***root*** *to enable mail delivery for other hosts on the network:*

- *Edit the* ***/etc/postfix/main.cf*** *file with a text editor, such as* ***vi****.*

- *Uncomment the* ***mydomain*** *line by removing the hash sign (* ***#****), and replace* ***domain.tld*** *with the domain the mail server is servicing, such as* ***example.com****.*

- *Uncomment the* ***myorigin = $mydomain*** *line.*

- *Uncomment the* ***myhostname*** *line, and replace* ***host.domain.tld*** *with the host name for the machine.*

- *Uncomment the* ***mydestination = $myhostname, localhost.$mydomain*** *line.*

- *Uncomment the* ***mynetworks*** *line, and replace* ***168.100.189.0/28*** *with a valid network setting for hosts that can connect to the server.*

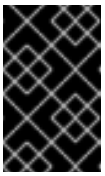
- *Uncomment the* ***inet_interfaces = all*** *line.*

- *Comment the* ***inet_interfaces = localhost*** *line.*

- *Restart the* ***postfix*** *service.*

Once these steps are complete, the host accepts outside emails for delivery.

Postfix has a large assortment of configuration options. One of the best ways to learn how to configure Postfix is to read the comments within the ***/etc/postfix/main.cf*** *configuration file. Additional resources including information about Postfix configuration, SpamAssassin integration, or detailed descriptions of the* ***/etc/postfix/main.cf*** *parameters are available online at* http://www.postfix.org/.



IMPORTANT

Due to the vulnerability described in POODLE: SSLv3 vulnerability (CVE-2014-3566)*, Red Hat recommends disabling* ***SSL*** *and using only* ***TLSv1.1*** *or* ***TLSv1.2****. See* Resolution for POODLE SSL 3.0 vulnerability (CVE-2014-3566) in Postfix and Dovecot *for details.*

15.3.1.4. Using Postfix with LDAP

Postfix can use an ***LDAP*** *directory as a source for various lookup tables (for example,* ***aliases****,* ***virtual****,* ***canonical****, and so on). This allows* ***LDAP*** *to store hierarchical user information and Postfix to only be given the result of* ***LDAP*** *queries when needed. By not storing this information locally, administrators can easily maintain it.*

15.3.1.4.1. The /etc/aliases lookup example

*The following is a basic example for using **LDAP** to look up the **/etc/aliases** file. Make sure your **/etc/postfix/main.cf** file contains the following:*

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

*Create a **/etc/postfix/ldap-aliases.cf** file if you do not have one already and make sure it contains the following:*

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

*where **ldap.example.com**, **example**, and **com** are parameters that need to be replaced with specification of an existing available **LDAP** server.*

NOTE

*The **/etc/postfix/ldap-aliases.cf** file can specify various parameters, including parameters that enable **LDAP SSL** and **STARTTLS**. For more information, see the **ldap_table(5)** man page.*

*For more information on **LDAP**, see OpenLDAP in the System-Level Authentication Guide.*

15.3.2. Sendmail

*Sendmail's core purpose, like other MTAs, is to safely transfer email between hosts, usually using the **SMTP** protocol. Note that Sendmail is considered deprecated and administrators are encouraged to use Postfix when possible. See Section 15.3.1, "Postfix" for more information.*

15.3.2.1. Purpose and Limitations

*It is important to be aware of what Sendmail is and what it can do, as opposed to what it is not. In these days of monolithic applications that fulfill multiple roles, Sendmail may seem like the only application needed to run an email server within an organization. Technically, this is true, as Sendmail can spool mail to each users' directory and deliver outbound mail for users. However, most users actually require much more than simple email delivery. Users usually want to interact with their email using an MUA, that uses **POP** or **IMAP**, to download their messages to their local machine. Or, they may prefer a Web interface to gain access to their mailbox. These other applications can work in conjunction with Sendmail, but they actually exist for different reasons and can operate separately from one another.*

It is beyond the scope of this section to go into all that Sendmail should or could be configured to do. With literally hundreds of different options and rule sets, entire volumes have been dedicated to helping explain everything that can be done and how to fix things that go wrong. See the Section 15.7, "Additional Resources" for a list of Sendmail resources.

This section reviews the files installed with Sendmail by default and reviews basic configuration changes, including how to stop unwanted email (spam) and how to extend Sendmail with the Lightweight Directory Access Protocol (LDAP).

15.3.2.2. The Default Sendmail Installation

*In order to use Sendmail, first ensure the **sendmail** package is installed on your system by running, as **root**:*

```
~]# yum install sendmail
```

*In order to configure Sendmail, ensure the **sendmail-cf** package is installed on your system by running, as **root**:*

```
~]# yum install sendmail-cf
```

For more information on installing packages with Yum, see Section 9.2.4, "Installing Packages".

Before using Sendmail, the default MTA has to be switched from Postfix. For more information how to switch the default MTA refer to Section 15.3, "Mail Transport Agents".

*The Sendmail executable is **sendmail**.*

*Sendmail configuration file is located at **/etc/mail/sendmail.cf**. Avoid editing the **sendmail.cf** file directly. To make configuration changes to Sendmail, edit the **/etc/mail/sendmail.mc** file, back up the original **/etc/mail/sendmail.cf** file, and restart the **sendmail** service. As a part of the restart, the sendmail.cf file and all binary representations of the databases are rebuild:*

```
# systemctl restart sendmail
```

More information on configuring Sendmail can be found in Section 15.3.2.3, "Common Sendmail Configuration Changes".

*Various Sendmail configuration files are installed in the **/etc/mail/** directory including:*

- ***access** – Specifies which systems can use Sendmail for outbound email.*
- ***domaintable** – Specifies domain name mapping.*
- ***local-host-names** – Specifies aliases for the host.*
- ***mailertable** – Specifies instructions that override routing for particular domains.*
- ***virtusertable** – Specifies a domain-specific form of aliasing, allowing multiple virtual domains to be hosted on one machine.*

*Several configuration files in the **/etc/mail/** directory, such as **access**, **domaintable**, **mailertable** and **virtusertable**, store their information in database files before Sendmail can use any configuration changes.*

To include any changes made to these configurations in their database files, run the following command:

```
# systemctl restart sendmail
```

15.3.2.3. Common Sendmail Configuration Changes

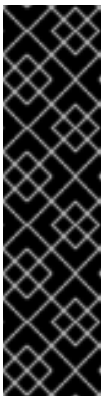
*When altering the Sendmail configuration file, it is best not to edit an existing file, but to generate an entirely new **/etc/mail/sendmail.cf** file.*

WARNING

*Before replacing or making any changes to the **sendmail.cf** file, create a backup copy.*

*To add the desired functionality to Sendmail, edit the **/etc/mail/sendmail.mc** file as **root**. Once you are finished, restart the **sendmail** service and, if the **m4** package is installed, the **m4** macro processor will automatically generate a new **sendmail.cf** configuration file:*

```
~]# systemctl restart sendmail
```



IMPORTANT

*The default **sendmail.cf** file does not allow Sendmail to accept network connections from any host other than the local computer. To configure Sendmail as a server for other clients, edit the **/etc/mail/sendmail.mc** file, and either change the address specified in the **Addr=** option of the **DAEMON_OPTIONS** directive from **127.0.0.1** to the IP address of an active network device or comment out the **DAEMON_OPTIONS** directive all together by placing **dnl** at the beginning of the line. When finished, regenerate **/etc/mail/sendmail.cf** by restarting the service:*

```
~]# systemctl restart sendmail
```

*The default configuration in Red Hat Enterprise Linux works for most **SMTP**-only sites.*

*Consult the **/usr/share/sendmail-cf/README** file before editing any files in the directories under the **/usr/share/sendmail-cf/** directory, as they can affect the future configuration of the **/etc/mail/sendmail.cf** file.*

15.3.2.4. Masquerading

*One common Sendmail configuration is to have a single machine act as a mail gateway for all machines on the network. For example, a company may want to have a machine called **mail.example.com** that handles all of their email and assigns a consistent return address to all outgoing mail.*

*In this situation, the Sendmail server must masquerade the machine names on the company network so that their return address is **user@example.com** instead of **user@host.example.com**.*

*To do this, add the following lines to **/etc/mail/sendmail.mc**:*

```
FEATURE(always_add_domain)dnl
FEATURE(masquerade_entire_domain)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(allmasquerade)dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(`example.com')dnl
```

*After generating a new **sendmail.cf** file from the changed configuration in sendmail.mc, restart the sendmail service by a following command:*

```
# systemctl restart sendmail
```

Note that administrators of mail servers, **DNS** and **DHCP** servers, as well as any provisioning applications, should agree on the host name format used in an organization. See the Red Hat Enterprise Linux 7 Networking Guide for more information on recommended naming practices.

15.3.2.5. Stopping Spam

Email spam can be defined as unnecessary and unwanted email received by a user who never requested the communication. It is a disruptive, costly, and widespread abuse of Internet communication standards.

Sendmail makes it relatively easy to block new spamming techniques being employed to send junk email. It even blocks many of the more usual spamming methods by default. Main anti-spam features available in sendmail are header checks, relaying denial (default from version 8.9), access database and sender information checks.

For example, forwarding of **SMTP** messages, also called relaying, has been disabled by default since Sendmail version 8.9. Before this change occurred, Sendmail directed the mail host (**x.edu**) to accept messages from one party (**y.com**) and sent them to a different party (**z.net**). Now, however, Sendmail must be configured to permit any domain to relay mail through the server. To configure relay domains, edit the **/etc/mail/relay-domains** file and restart Sendmail

```
~]# systemctl restart sendmail
```

However, servers on the Internet can also send spam messages. In these instances, Sendmail's access control features available through the **/etc/mail/access** file can be used to prevent connections from unwanted hosts. The following example illustrates how this file can be used to both block and specifically allow access to the Sendmail server:

```
badspammer.com ERROR:550 "Go away and do not spam us anymore" tux.badspammer.com
OK 10.0 RELAY
```

This example shows that any email sent from **badspammer.com** is blocked with a 550 RFC-821 compliant error code, with a message sent back. Emails sent from the **tux.badspammer.com** sub-domain are accepted. The last line shows that any email sent from the 10.0.. network can be relayed through the mail server.

Because the **/etc/mail/access.db** file is a database, use the following command to update any changes:

```
# systemctl restart sendmail
```

The above examples only represent a small part of what Sendmail can do in terms of allowing or blocking access. See the **/usr/share/sendmail-cf/README** file for more information and examples.

Since Sendmail calls the Procmail MDA when delivering mail, it is also possible to use a spam filtering program, such as SpamAssassin, to identify and file spam for users. See Section 15.4.2.6, "Spam Filters" for more information about using SpamAssassin.

15.3.2.6. Using Sendmail with LDAP

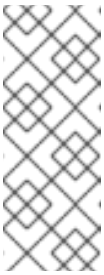
Using **LDAP** is a very quick and powerful way to find specific information about a particular user from a much larger group. For example, an **LDAP** server can be used to look up a particular email address from a common corporate directory by the user's last name. In this kind of implementation, **LDAP** is largely

*separate from Sendmail, with **LDAP** storing the hierarchical user information and Sendmail only being given the result of **LDAP** queries in pre-addressed email messages.*

*However, Sendmail supports a much greater integration with **LDAP**, where it uses **LDAP** to replace separately maintained files, such as **/etc/aliases** and **/etc/mail/virtusertables**, on different mail servers that work together to support a medium- to enterprise-level organization. In short, **LDAP** abstracts the mail routing level from Sendmail and its separate configuration files to a powerful **LDAP** cluster that can be leveraged by many different applications.*

*The current version of Sendmail contains support for **LDAP**. To extend the Sendmail server using **LDAP**, first get an **LDAP** server, such as **OpenLDAP**, running and properly configured. Then edit the **/etc/mail/sendmail.mc** to include the following:*

```
LDAPROUTE_DOMAIN('yourdomain.com')dnl
FEATURE('ldap_routing')dnl
```



NOTE

*This is only for a very basic configuration of Sendmail with **LDAP**. The configuration can differ greatly from this depending on the implementation of **LDAP**, especially when configuring several Sendmail machines to use a common **LDAP** server.*

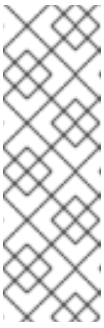
*Consult **/usr/share/sendmail-cf/README** for detailed **LDAP** routing configuration instructions and examples.*

*Next, recreate the **/etc/mail/sendmail.cf** file by running the **m4** macro processor and again restarting Sendmail. See Section 15.3.2.3, "Common Sendmail Configuration Changes" for instructions.*

*For more information on **LDAP**, see OpenLDAP in the System-Level Authentication Guide.*

15.3.3. Fetchmail

*Fetchmail is an MTA which retrieves email from remote servers and delivers it to the local MTA. Many users appreciate the ability to separate the process of downloading their messages located on a remote server from the process of reading and organizing their email in an MUA. Designed with the needs of dial-up users in mind, Fetchmail connects and quickly downloads all of the email messages to the mail spool file using any number of protocols, including **POP3** and **IMAP**. It can even forward email messages to an **SMTP** server, if necessary.*



NOTE

*In order to use **Fetchmail**, first ensure the **fetchmail** package is installed on your system by running, as **root**:*

```
~]# yum install fetchmail
```

For more information on installing packages with Yum, see Section 9.2.4, "Installing Packages".

*Fetchmail is configured for each user through the use of a **.fetchmailrc** file in the user's home directory. If it does not already exist, create the **.fetchmailrc** file in your home directory*

*Using preferences in the **.fetchmailrc** file, Fetchmail checks for email on a remote server and downloads*

it. It then delivers it to port 25 on the local machine, using the local MTA to place the email in the correct user's spool file. If Procmail is available, it is launched to filter the email and place it in a mailbox so that it can be read by an MUA.

15.3.3.1. Fetchmail Configuration Options

Although it is possible to pass all necessary options on the command line to check for email on a remote server when executing Fetchmail, using a ***.fetchmailrc*** *file is much easier. Place any desired configuration options in the* ***.fetchmailrc*** *file for those options to be used each time the* ***fetchmail*** *command is issued. It is possible to override these at the time Fetchmail is run by specifying that option on the command line.*

A user's ***.fetchmailrc*** *file contains three classes of configuration options:*

- *global options — Gives Fetchmail instructions that control the operation of the program or provide settings for every connection that checks for email.*
- *server options — Specifies necessary information about the server being polled, such as the host name, as well as preferences for specific email servers, such as the port to check or number of seconds to wait before timing out. These options affect every user using that server.*
- *user options — Contains information, such as user name and password, necessary to authenticate and check for email using a specified email server.*

Global options appear at the top of the ***.fetchmailrc*** *file, followed by one or more server options, each of which designate a different email server that Fetchmail should check. User options follow server options for each user account checking that email server. Like server options, multiple user options may be specified for use with a particular server as well as to check multiple email accounts on the same server.*

Server options are called into service in the ***.fetchmailrc*** *file by the use of a special option verb,* ***poll*** *or* ***skip****, that precedes any of the server information. The* ***poll*** *action tells Fetchmail to use this server option when it is run, which checks for email using the specified user options. Any server options after a* ***skip*** *action, however, are not checked unless this server's host name is specified when Fetchmail is invoked. The* ***skip*** *option is useful when testing configurations in the* ***.fetchmailrc*** *file because it only checks skipped servers when specifically invoked, and does not affect any currently working configurations.*

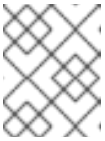
The following is an example of a ***.fetchmailrc*** *file:*

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
  user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
  user 'user5' there with password 'secret2' is user1 here
  user 'user7' there with password 'secret3' is user1 here
```

*In this example, the global options specify that the user is sent email as a last resort (****postmaster*** *option) and all email errors are sent to the postmaster instead of the sender (****bouncemail*** *option). The* ***set*** *action tells Fetchmail that this line contains a global option. Then, two email servers are specified, one set to check using* ***POP3****, the other for trying various protocols to find one that works. Two users are checked using the second server option, but all email found for any user is sent to* ***user1****'s mail spool. This allows multiple mailboxes to be checked on multiple servers, while appearing in a single MUA inbox. Each user's specific information begins with the* ***user*** *action.*



NOTE

Users are not required to place their password in the ***.fetchmailrc*** *file. Omitting the* ***with password 'password'*** *section causes Fetchmail to ask for a password when it is launched.*

Fetchmail has numerous global, server, and local options. Many of these options are rarely used or only apply to very specific situations. The ***fetchmail*** *man page explains each option in detail, but the most common ones are listed in the following three sections.*

15.3.3.2. Global Options

Each global option should be placed on a single line after a ***set*** *action.*

- ***daemon seconds*** *– Specifies daemon-mode, where Fetchmail stays in the background. Replace seconds with the number of seconds Fetchmail is to wait before polling the server.*
- ***postmaster*** *– Specifies a local user to send mail to in case of delivery problems.*
- ***syslog*** *– Specifies the log file for errors and status messages. By default, this is* ***/var/log/maillog****.*

15.3.3.3. Server Options

Server options must be placed on their own line in ***.fetchmailrc*** *after a* ***poll*** *or* ***skip*** *action.*

- ***auth auth-type*** *– Replace auth-type with the type of authentication to be used. By default,* ***password*** *authentication is used, but some protocols support other types of authentication, including* ***kerberos_v5****,* ***kerberos_v4****, and* ***ssh****. If the* ***any*** *authentication type is used, Fetchmail first tries methods that do not require a password, then methods that mask the password, and finally attempts to send the password unencrypted to authenticate to the server.*
- ***interval number*** *– Polls the specified server every* ***number*** *of times that it checks for email on all configured servers. This option is generally used for email servers where the user rarely receives messages.*
- ***port port-number*** *– Replace port-number with the port number. This value overrides the default port number for the specified protocol.*
- ***proto protocol*** *– Replace protocol with the protocol, such as* ***pop3*** *or* ***imap****, to use when checking for messages on the server.*
- ***timeout seconds*** *– Replace seconds with the number of seconds of server inactivity after which Fetchmail gives up on a connection attempt. If this value is not set, a default of* ***300*** *seconds is used.*

15.3.3.4. User Options

User options may be placed on their own lines beneath a server option or on the same line as the server option. In either case, the defined options must follow the ***user*** *option (defined below).*

- ***fetchall*** *– Orders Fetchmail to download all messages in the queue, including messages that have already been viewed. By default, Fetchmail only pulls down new messages.*
- ***fetchlimit number*** *– Replace number with the number of messages to be retrieved before stopping.*
- ***flush*** *– Deletes all previously viewed messages in the queue before retrieving new messages.*

- ***limit max-number-bytes*** – Replace *max-number-bytes* with the maximum size in bytes that messages are allowed to be when retrieved by Fetchmail. This option is useful with slow network links, when a large message takes too long to download.

- ***password 'password'*** – Replace *password* with the user's password.

- ***preconnect "command"*** – Replace *command* with a command to be executed before retrieving messages for the user.

- ***postconnect "command"*** – Replace *command* with a command to be executed after retrieving messages for the user.

- ***ssl*** – Activates SSL encryption. At the time of writing, the default action is to use the best available from ***SSL2***, ***SSL3***, ***SSL23***, ***TLS1***, ***TLS1.1*** and ***TLS1.2***. Note that ***SSL2*** is considered obsolete and due to the POODLE: SSLv3 vulnerability (CVE-2014-3566), ***SSLv3*** should not be used. However there is no way to force the use of TLS1 or newer, therefore ensure the mail server being connected to is configured not to use ***SSLv2*** and ***SSLv3***. Use ***stunnel*** where the server cannot be configured not to use ***SSLv2*** and ***SSLv3***.

- ***sslproto*** – Defines allowed SSL or TLS protocols. Possible values are ***SSL2***, ***SSL3***, ***SSL23***, and ***TLS1***. The default value, if ***sslproto*** is omitted, unset, or set to an invalid value, is ***SSL23***. The default action is to use the best from ***SSLv2***, ***SSLv3***, ***TLSv1***, ***TLS1.1*** and ***TLS1.2***. Note that setting any other value for SSL or TLS will disable all the other protocols. Due to the POODLE: SSLv3 vulnerability (CVE-2014-3566), it is recommend to omit this option, or set it to ***SSLv23***, and configure the corresponding mail server not to use ***SSLv2*** and ***SSLv3***. Use ***stunnel*** where the server cannot be configured not to use ***SSLv2*** and ***SSLv3***.

- ***user "username"*** – Replace *username* with the user name used by Fetchmail to retrieve messages. This option must precede all other user options.

15.3.3.5. Fetchmail Command Options

Most Fetchmail options used on the command line when executing the ***fetchmail*** command mirror the ***.fetchmailrc*** configuration options. In this way, Fetchmail may be used with or without a configuration file. These options are not used on the command line by most users because it is easier to leave them in the ***.fetchmailrc*** file.

There may be times when it is desirable to run the ***fetchmail*** command with other options for a particular purpose. It is possible to issue command options to temporarily override a ***.fetchmailrc*** setting that is causing an error, as any options specified at the command line override configuration file options.

15.3.3.6. Informational or Debugging Options

Certain options used after the ***fetchmail*** command can supply important information.

- ***--configdump*** – Displays every possible option based on information from ***.fetchmailrc*** and Fetchmail defaults. No email is retrieved for any users when using this option.

- ***-s*** – Executes Fetchmail in silent mode, preventing any messages, other than errors, from appearing after the ***fetchmail*** command.

- ***-v*** – Executes Fetchmail in verbose mode, displaying every communication between Fetchmail and remote email servers.

- ***-V*** *– Displays detailed version information, lists its global options, and shows settings to be used with each user, including the email protocol and authentication method. No email is retrieved for any users when using this option.*

15.3.3.7. Special Options

These options are occasionally useful for overriding defaults often found in the ***.fetchmailrc*** *file.*

- ***-a*** *– Fetchmail downloads all messages from the remote email server, whether new or previously viewed. By default, Fetchmail only downloads new messages.*
- ***-k*** *– Fetchmail leaves the messages on the remote email server after downloading them. This option overrides the default behavior of deleting messages after downloading them.*
- ***-l max-number-bytes*** *– Fetchmail does not download any messages over a particular size and leaves them on the remote email server.*
- ***--quit*** *– Quits the Fetchmail daemon process.*

More commands and ***.fetchmailrc*** *options can be found in the* ***fetchmail*** *man page.*

15.3.4. Mail Transport Agent (MTA) Configuration

A Mail Transport Agent (MTA) is essential for sending email. A Mail User Agent (MUA) such as ***Evolution*** *or* ***Mutt****, is used to read and compose email. When a user sends an email from an MUA, the message is handed off to the MTA, which sends the message through a series of MTAs until it reaches its destination.*

Even if a user does not plan to send email from the system, some automated tasks or system programs might use the ***mail*** *command to send email containing log messages to the* ***root*** *user of the local system.*

Red Hat Enterprise Linux 7 provides two MTAs: Postfix and Sendmail. If both are installed, Postfix is the default MTA.

15.4. MAIL DELIVERY AGENTS

Red Hat Enterprise Linux includes ***Procmail*** *as primary MDA. Both applications are considered LDAs and both move email from the MTA's spool file into the user's mailbox. However, Procmail provides a robust filtering system.*

This section details only Procmail. For information on the ***mail*** *command, consult its man page (* ***man mail****).*

Procmail delivers and filters email as it is placed in the mail spool file of the localhost. It is powerful, gentle on system resources, and widely used. Procmail can play a critical role in delivering email to be read by email client applications.

Procmail can be invoked in several different ways. Whenever an MTA places an email into the mail spool file, Procmail is launched. Procmail then filters and files the email for the MUA and quits. Alternatively, the MUA can be configured to execute Procmail any time a message is received so that messages are moved into their correct mailboxes. By default, the presence of ***/etc/procmailrc*** *or of a* ***~/.procmailrc*** *file (also called an rc file) in the user's home directory invokes Procmail whenever an MTA receives a new message.*

By default, no system-wide ***rc*** *files exist in the* ***/etc*** *directory and no* ***.procmailrc*** *files exist in any user's home directory. Therefore, to use Procmail, each user must construct a* ***.procmailrc*** *file with specific environment variables and rules.*

Whether Procmail acts upon an email message depends upon whether the message matches a specified set of conditions or recipes in the ***rc*** *file. If a message matches a recipe, then the email is placed in a specified file, is deleted, or is otherwise processed.*

When Procmail starts, it reads the email message and separates the body from the header information. Next, Procmail looks for a ***/etc/procmailrc*** *file and* ***rc*** *files in the* ***/etc/procmailrcs/*** *directory for default, system-wide, Procmail environmental variables and recipes. Procmail then searches for a* ***.procmailrc*** *file in the user's home directory. Many users also create additional* ***rc*** *files for Procmail that are referred to within the* ***.procmailrc*** *file in their home directory.*

15.4.1. Procmail Configuration

The Procmail configuration file contains important environmental variables. These variables specify things such as which messages to sort and what to do with the messages that do not match any recipes.

*These environmental variables usually appear at the beginning of the ~/****.procmailrc*** *file in the following format:*

```
env-variable="value"
```

In this example, ***env-variable*** *is the name of the variable and* ***value*** *defines the variable.*

There are many environment variables not used by most Procmail users and many of the more important environment variables are already defined by a default value. Most of the time, the following variables are used:

- ***DEFAULT*** *– Sets the default mailbox where messages that do not match any recipes are placed. The default* ***DEFAULT*** *value is the same as* ***$ORGMAIL****.*

- ***INCLUDERC*** *– Specifies additional* ***rc*** *files containing more recipes for messages to be checked against. This breaks up the Procmail recipe lists into individual files that fulfill different roles, such as blocking spam and managing email lists, that can then be turned off or on by using comment characters in the user's ~/****.procmailrc*** *file.*
  *For example, lines in a user's ~/****.procmailrc*** *file may look like this:*

  ```
  MAILDIR=$HOME/Msgs
  INCLUDERC=$MAILDIR/lists.rc
  INCLUDERC=$MAILDIR/spam.rc
  ```

  To turn off Procmail filtering of email lists but leaving spam control in place, comment out the first ***INCLUDERC*** *line with a hash sign (****#****). Note that it uses paths relative to the current directory.*

- ***LOCKSLEEP*** *– Sets the amount of time, in seconds, between attempts by Procmail to use a particular lockfile. The default is 8 seconds.*

- ***LOCKTIMEOUT*** *– Sets the amount of time, in seconds, that must pass after a lockfile was last modified before Procmail assumes that the lockfile is old and can be deleted. The default is 1024 seconds.*

- ***LOGFILE*** *– The file to which any Procmail information or error messages are written.*

- ***MAILDIR*** *– Sets the current working directory for Procmail. If set, all other Procmail paths are relative to this directory.*

- ***ORGMAIL*** – *Specifies the original mailbox, or another place to put the messages if they cannot be placed in the default or recipe-required location.*
  *By default, a value of /**var**/**spool**/**mail**/**$LOGNAME** is used.*

- ***SUSPEND*** – *Sets the amount of time, in seconds, that Procmail pauses if a necessary resource, such as swap space, is not available.*

- ***SWITCHRC*** – *Allows a user to specify an external file containing additional Procmail recipes, much like the **INCLUDERC** option, except that recipe checking is actually stopped on the referring configuration file and only the recipes on the **SWITCHRC**-specified file are used.*

- ***VERBOSE*** – *Causes Procmail to log more information. This option is useful for debugging.*

*Other important environmental variables are pulled from the shell, such as **LOGNAME**, the login name; **HOME**, the location of the home directory; and **SHELL**, the default shell.*

*A comprehensive explanation of all environments variables, and their default values, is available in the **procmailrc** man page.*

15.4.2. Procmail Recipes

New users often find the construction of recipes the most difficult part of learning to use Procmail. This difficulty is often attributed to recipes matching messages by using regular expressions which are used to specify qualifications for string matching. However, regular expressions are not very difficult to construct and even less difficult to understand when read. Additionally, the consistency of the way Procmail recipes are written, regardless of regular expressions, makes it easy to learn by example. To see example Procmail recipes, see Section 15.4.2.5, "Recipe Examples".

Procmail recipes take the following form:

```
:0 flags : lockfile-name
* condition_1_special-condition-character condition_1_regular_expression
* condition_2_special-condition-character condition-2_regular_expression
* condition_N_special-condition-character condition-N_regular_expression
  special-action-character
  action-to-perform
```

*The first two characters in a Procmail recipe are a colon and a zero. Various flags can be placed after the zero to control how Procmail processes the recipe. A colon after the **flags** section specifies that a lockfile is created for this message. If a lockfile is created, the name can be specified by replacing **lockfile-name**.*

A recipe can contain several conditions to match against the message. If it has no conditions, every message matches the recipe. Regular expressions are placed in some conditions to facilitate message matching. If multiple conditions are used, they must all match for the action to be performed. Conditions are checked based on the flags set in the recipe's first line. Optional special characters placed after the asterisk character (*) can further control the condition.

*The **action-to-perform** argument specifies the action taken when the message matches one of the conditions. There can only be one action per recipe. In many cases, the name of a mailbox is used here to direct matching messages into that file, effectively sorting the email. Special action characters may also be used before the action is specified. See Section 15.4.2.4, "Special Conditions and Actions" for more information.*

15.4.2.1. Delivering vs. Non-Delivering Recipes

The action used if the recipe matches a particular message determines whether it is considered a

*delivering or non-delivering recipe. A delivering recipe contains an action that writes the message to a file, sends the message to another program, or forwards the message to another email address. A non-delivering recipe covers any other actions, such as a nesting block. A nesting block is a set of actions, contained in braces **{}**, that are performed on messages which match the recipe's conditions. Nesting blocks can be nested inside one another, providing greater control for identifying and performing actions on messages.*

When messages match a delivering recipe, Procmail performs the specified action and stops comparing the message against any other recipes. Messages that match non-delivering recipes continue to be compared against other recipes.

15.4.2.2. Flags

*Flags are essential to determine how or if a recipe's conditions are compared to a message. The **egrep** utility is used internally for matching of the conditions. The following flags are commonly used:*

- ***A** – Specifies that this recipe is only used if the previous recipe without an **A** or **a** flag also matched this message.*
- ***a** – Specifies that this recipe is only used if the previous recipe with an **A** or **a** flag also matched this message and was successfully completed.*
- ***B** – Parses the body of the message and looks for matching conditions.*
- ***b** – Uses the body in any resulting action, such as writing the message to a file or forwarding it. This is the default behavior.*
- ***c** – Generates a carbon copy of the email. This is useful with delivering recipes, since the required action can be performed on the message and a copy of the message can continue being processed in the **rc** files.*
- ***D** – Makes the **egrep** comparison case-sensitive. By default, the comparison process is not case-sensitive.*
- ***E** – While similar to the **A** flag, the conditions in the recipe are only compared to the message if the immediately preceding recipe without an **E** flag did not match. This is comparable to an **else** action.*
- ***e** – The recipe is compared to the message only if the action specified in the immediately preceding recipe fails.*
- ***f** – Uses the pipe as a filter.*
- ***H** – Parses the header of the message and looks for matching conditions. This is the default behavior.*
- ***h** – Uses the header in a resulting action. This is the default behavior.*
- ***w** – Tells Procmail to wait for the specified filter or program to finish, and reports whether or not it was successful before considering the message filtered.*
- ***W** – Is identical to **w** except that "Program failure" messages are suppressed.*

*For a detailed list of additional flags, see the **procmailrc** man page.*

15.4.2.3. Specifying a Local Lockfile

*Lockfiles are very useful with Procmail to ensure that more than one process does not try to alter a message simultaneously. Specify a local lockfile by placing a colon (**:**) after any flags on a recipe's first line. This creates a local lockfile based on the destination file name plus whatever has been set in the **LOCKEXT** global environment variable.*

Alternatively, specify the name of the local lockfile to be used with this recipe after the colon.

15.4.2.4. Special Conditions and Actions

Special characters used before Procmail recipe conditions and actions change the way they are interpreted.

The following characters may be used after the asterisk character (*****) at the beginning of a recipe's condition line:

- ***!** – In the condition line, this character inverts the condition, causing a match to occur only if the condition does not match the message.*
- ***<** – Checks if the message is under a specified number of bytes.*
- ***>** – Checks if the message is over a specified number of bytes.*

The following characters are used to perform special actions:

- ***!** – In the action line, this character tells Procmail to forward the message to the specified email addresses.*
- ***$** – Refers to a variable set earlier in the **rc** file. This is often used to set a common mailbox that is referred to by various recipes.*
- *| – Starts a specified program to process the message.*
- ***{** and **}** – Constructs a nesting block, used to contain additional recipes to apply to matching messages.*

If no special character is used at the beginning of the action line, Procmail assumes that the action line is specifying the mailbox in which to write the message.

15.4.2.5. Recipe Examples

Procmail is an extremely flexible program, but as a result of this flexibility, composing Procmail recipes from scratch can be difficult for new users.

*The best way to develop the skills to build Procmail recipe conditions stems from a strong understanding of regular expressions combined with looking at many examples built by others. A thorough explanation of regular expressions is beyond the scope of this section. The structure of Procmail recipes and useful sample Procmail recipes can be found at various places on the Internet. The proper use and adaptation of regular expressions can be derived by viewing these recipe examples. In addition, introductory information about basic regular expression rules can be found in the **grep(1)** man page.*

The following simple examples demonstrate the basic structure of Procmail recipes and can provide the foundation for more intricate constructions.

A basic recipe may not even contain conditions, as is illustrated in the following example:

```
:0:
new-mail.spool
```

The first line specifies that a local lockfile is to be created but does not specify a name, so Procmail uses the destination file name and appends the value specified in the ***LOCKEXT*** *environment variable. No condition is specified, so every message matches this recipe and is placed in the single spool file called* ***new-mail.spool****, located within the directory specified by the* ***MAILDIR*** *environment variable. An MUA can then view messages in this file.*

A basic recipe, such as this, can be placed at the end of all ***rc*** *files to direct messages to a default location.*

The following example matched messages from a specific email address and throws them away.

```
:0
* ^From: spammer@domain.com
/dev/null
```

With this example, any messages sent by ***spammer@domain.com*** *are sent to the* ***/dev/null*** *device, deleting them.*

WARNING

Be certain that rules are working as intended before sending messages to ***/dev/null*** *for permanent deletion. If a recipe inadvertently catches unintended messages, and those messages disappear, it becomes difficult to troubleshoot the rule.*

A better solution is to point the recipe's action to a special mailbox, which can be checked from time to time to look for false positives. Once satisfied that no messages are accidentally being matched, delete the mailbox and direct the action to send the messages to ***/dev/null****.*

The following recipe grabs email sent from a particular mailing list and places it in a specified folder.

```
:0:
* ^(From|Cc|To).*tux-lug
tuxlug
```

Any messages sent from the ***tux-lug@domain.com*** *mailing list are placed in the* ***tuxlug*** *mailbox automatically for the MUA. Note that the condition in this example matches the message if it has the mailing list's email address on the* ***From****,* ***Cc****, or* ***To*** *lines.*

Consult the many Procmail online resources available in Section 15.7, "Additional Resources" for more detailed and powerful recipes.

15.4.2.6. Spam Filters

Because it is called by Sendmail, Postfix, and Fetchmail upon receiving new emails, Procmail can be used as a powerful tool for combating spam.

This is particularly true when Procmail is used in conjunction with SpamAssassin. When used together, these two applications can quickly identify spam emails, and sort or destroy them.

SpamAssassin uses header analysis, text analysis, blacklists, a spam-tracking database, and self-learning Bayesian spam analysis to quickly and accurately identify and tag spam.

NOTE

In order to use ***SpamAssassin****, first ensure the* ***spamassassin*** *package is installed on your system by running, as* ***root***

```
~]# yum install spamassassin
```

For more information on installing packages with Yum, see Section 9.2.4, "Installing Packages".

The easiest way for a local user to use SpamAssassin is to place the following line near the top of the ***~/.procmailrc*** *file:*

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

The ***/etc/mail/spamassassin/spamassassin-default.rc*** *contains a simple Procmail rule that activates SpamAssassin for all incoming email. If an email is determined to be spam, it is tagged in the header as such and the title is prepended with the following pattern:*

```
SPAM
```

The message body of the email is also prepended with a running tally of what elements caused it to be diagnosed as spam.

To file email tagged as spam, a rule similar to the following can be used:

```
:0 Hw
* ^X-Spam-Status: Yes
spam
```

This rule files all email tagged in the header as spam into a mailbox called* spam*.

Since SpamAssassin is a Perl script, it may be necessary on busy servers to use the binary SpamAssassin daemon (spamd) and the client application (spamc). Configuring SpamAssassin this way, however, requires root access to the host.

To start the* spamd *daemon, type the following command:

```
~]# systemctl start spamassassin
```

To start the SpamAssassin daemon when the system is booted, run:

```
systemctl enable spamassassin.service
```

See Chapter 10, Managing Services with systemd for more information about starting and stopping services.

To configure Procmail to use the SpamAssassin client application instead of the Perl script, place the following line near the top of the ~/**.procmailrc** file. For a system-wide configuration, place it in **/etc/procmailrc**:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

15.5. MAIL USER AGENTS

Red Hat Enterprise Linux offers a variety of email programs, both graphical email client programs, such as Evolution, and text-based email programs such as**mutt**.

The remainder of this section focuses on securing communication between a client and a server.

15.5.1. Securing Communication

MUAs included with Red Hat Enterprise Linux, such as Thunderbird, Evolution and Mutt offer SSL-encrypted email sessions.

Like any other service that flows over a network unencrypted, important email information, such as user names, passwords, and entire messages, may be intercepted and viewed by users on the network. Additionally, since the standard **POP** and **IMAP** protocols pass authentication information unencrypted, it is possible for an attacker to gain access to user accounts by collecting user names and passwords as they are passed over the network.

15.5.1.1. Secure Email Clients

Most Linux MUAs designed to check email on remote servers support SSL encryption. To use SSL when retrieving email, it must be enabled on both the email client and the server.

SSL is easy to enable on the client-side, often done with the click of a button in the MUA's configuration window or via an option in the MUA's configuration file. Secure **IMAP** and **POP** have known port numbers (993 and 995, respectively) that the MUA uses to authenticate and download messages.

15.5.1.2. Securing Email Client Communications

Offering SSL encryption to **IMAP** and **POP** users on the email server is a simple matter.

First, create an SSL certificate. This can be done in two ways: by applying to a Certificate Authority (CA) for an SSL certificate or by creating a self-signed certificate.

WARNING

Self-signed certificates should be used for testing purposes only. Any server used in a production environment should use an SSL certificate signed by a CA.

To create a self-signed SSL certificate for **IMAP** or **POP**, change to the **/etc/pki/dovecot/** directory, edit the certificate parameters in the **/etc/pki/dovecot/dovecot-openssl.cnf** configuration file as you prefer, and type the following commands, as **root**:

```
dovecot]# rm -f certs/dovecot.pem private/dovecot.pem
dovecot]# /usr/libexec/dovecot/mkcert.sh
```

Once finished, make sure you have the following configurations in your **/etc/dovecot/conf.d/10-ssl.conf** file:

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

Issue the following command to restart the **dovecot** daemon:

```
~]# systemctl restart dovecot
```

Alternatively, the **stunnel** command can be used as an encryption wrapper around the standard, non-secure connections to **IMAP** or **POP** services.

The **stunnel** utility uses external OpenSSL libraries included with Red Hat Enterprise Linux to provide strong cryptography and to protect the network connections. It is recommended to apply to a CA to obtain an SSL certificate, but it is also possible to create a self-signed certificate.

See Using stunnel in the Red Hat Enterprise Linux 7 Security Guide for instructions on how to install **stunnel** and create its basic configuration. To configure **stunnel** as a wrapper for **IMAPS** and **POP3S**, add the following lines to the **/etc/stunnel/stunnel.conf** configuration file:

```
[pop3s]
accept = 995
connect = 110

[imaps]
accept = 993
connect = 143
```

The Security Guide also explains how to start and stop **stunnel**. Once you start it, it is possible to use an **IMAP** or a **POP** email client and connect to the email server using SSL encryption.

15.6. CONFIGURING MAIL SERVER WITH ANTISPAM AND ANTIVIRUS

Once your email delivery works, incoming emails may contain unsolicited messages also known as spam. These messages can also contain harmful viruses and malware, posing security risk and potential production loss on your systems.

To avoid these risks, you can filter the incoming messages and check them against viruses by using an antispam and antivirus solution.

15.6.1. Configuring Spam Filtering for Mail Transport Agent or Mail Delivery Agent

You can filter spam in a Mail Transport Agent (MTA), Mail Delivery Agent (MDA), or Mail User Agent (MUA). This chapter describes spam filtering in MTAs and MDAs.

15.6.1.1. Configuring Spam Filtering in a Mail Transport Agent

Red Hat Enterprise Linux 7 offers two primary MTAs: Postfix and Sendmail.

For details on how to install and configure an MTA, see Section 15.3, "Mail Transport Agents".

Stopping spam in a MTA side is possible with the use of Sendmail, which has several anti-spam features: header checks, relaying denial, access database and sender information checks. Fore more information, see Section 15.3.2.5, "Stopping Spam".

Moreover, both Postfix and Sendmail can work with third-party mail filters (milters) to filter spam and viruses in the mail-processing chain. In case of Postfix, the support for milters is included directly in the postfix package. In case of Sendmail, you need to install the sendmail-milter package, to be able to use milters.

15.6.1.2. Configuring Spam Filtering in a Mail Delivery Agent

Red Hat Enterprise Linux includes two primary MDAs, Procmail and the **mail** utility. See Section 15.2.2, "Mail Delivery Agent" for more information.

To stop spam in an MDA, users of Procmail can install third-party software named SpamAssassin available in the spamassassin package. SpamAssassin is a spam detection system that uses a variety of methods to identify spam in incoming mail. For further information on Spamassassin installation, configuration and deployment, see Section 15.4.2.6, "Spam Filters" or the How can I configure Spamassassin to filter all the incoming mail on my server? Red Hat Knowledgebase article. For additional information on SpamAssassin, see the SpamAssassin project website.

WARNING

Note that SpamAssasin is a third-party software, and Red Hat does not support its use.

The spamassassin package is available only through the Extra Packages for Enterprise Linux (EPEL) repository. To learn more about using the EPEL repository, see Section 15.6.3, "Using the EPEL Repository to install Antispam and Antivirus Software".

To learn more about how Red Hat handles the third party software and what level of support for it Red Hat provides, see How does Red Hat Global Support Services handle third-party software, drivers, and/or uncertified hardware/hypervisors or guest operating systems? Red Hat Knowledgebase article.

15.6.2. Configuring Antivirus Protection

To protect your system against viruses, you can install ClamAV, an open source antivirus engine for detecting trojans, viruses, malware, and other malicious software. For additional information about ClamAV, see the ClamAV project website.

WARNING

Note that ClamAV is a third-party software, and Red Hat does not support its use.

The clamav, clamav-data, clamav-server and clamav-update packages are only available in the Extra Packages for Enterprise Linux (EPEL) repository. To learn more about using the EPEL repository, see Section 15.6.3, "Using the EPEL Repository to install Antispam and Antivirus Software".

To learn more about how Red Hat handles the third party software and what level of support for it Red Hat provides, see How does Red Hat Global Support Services handle third-party software, drivers, and/or uncertified hardware/hypervisors or guest operating systems? Red Hat Knowledgebase article.

*Once you have enabled the EPEL repository, install ClamAV by running the following command as the **root** user:*

```
~]# yum install clamav clamav-data clamav-server clamav-update
```

15.6.3. Using the EPEL Repository to install Antispam and Antivirus Software

EPEL is a Fedora Special Interest Group that creates, maintains, and manages a high quality set of additional packages for Red Hat Enterprise Linux. For more information, see the Fedora EPEL website.

*To use the EPEL repository, download the latest version of the epel-release package for Red Hat Enterprise Linux 7. You can also run the following command as the **root** user:*

```
~]# yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpmzu
```

When using the EPEL repository for the first time, you need to authenticate with a public GPG key. For more information, see Fedora Package Signing Keys.

15.7. ADDITIONAL RESOURCES

The following is a list of additional documentation about email applications.

15.7.1. Installed Documentation

- *Information on configuring Sendmail is included with the sendmail and sendmail-cf packages.*
  - */**usr**/**share**/**sendmail-cf**/**README** – Contains information on the **m4** macro processor, file locations for Sendmail, supported mailers, how to access enhanced features, and more. In addition, the **sendmail** and **aliases** man pages contain helpful information covering various Sendmail options and the proper configuration of the Sendmail /**etc**/**mail**/**aliases** file.*

- */usr/share/doc/postfix-version-number/* — Contains a large amount of information on how to configure Postfix. Replace version-number with the version number of Postfix.

- */usr/share/doc/fetchmail-version-number/* — Contains a full list of Fetchmail features in the **FEATURES** file and an introductory **FAQ** document. Replace version-number with the version number of Fetchmail.

- */usr/share/doc/procmail-version-number/* — Contains a **README** file that provides an overview of Procmail, a **FEATURES** file that explores every program feature, and an **FAQ** file with answers to many common configuration questions. Replace version-number with the version number of Procmail.
  When learning how Procmail works and creating new recipes, the following Procmail man pages are invaluable:

  - **procmail** — Provides an overview of how Procmail works and the steps involved with filtering email.

  - **procmailrc** — Explains the **rc** file format used to construct recipes.

  - **procmailex** — Gives a number of useful, real-world examples of Procmail recipes.

  - **procmailsc** — Explains the weighted scoring technique used by Procmail to match a particular recipe to a message.

  - */usr/share/doc/spamassassin-version-number/* — Contains a large amount of information pertaining to SpamAssassin. Replace version-number with the version number of the spamassassin package.

15.7.2. Online Documentation

- How to configure postfix with TLS? — A Red Hat Knowledgebase article that describes configuring postfix to use TLS.

- How to configure a Sendmail Smart Host — A Red Hat Knowledgebase solution that describes configuring a sendmail Smart Host.

- http://www.sendmail.org/ — Offers a thorough technical breakdown of Sendmail features, documentation and configuration examples.

- http://www.sendmail.com/ — Contains news, interviews and articles concerning Sendmail, including an expanded view of the many options available.

- http://www.postfix.org/ — The Postfix project home page contains a wealth of information about Postfix. The mailing list is a particularly good place to look for information.

- http://www.fetchmail.info/fetchmail-FAQ.html — A thorough FAQ about Fetchmail.

- http://www.spamassassin.org/ — The official site of the SpamAssassin project.

15.7.3. Related Books

- Sendmail Milters: A Guide for Fighting Spam by Bryan Costales and Marcia Flynt; Addison-Wesley — A good Sendmail guide that can help you customize your mail filters.

- Sendmail by Bryan Costales with Eric Allman et al.; O'Reilly & Associates — A good Sendmail reference written with the assistance of the original creator of Delivermail and Sendmail.

- *Removing the Spam: Email Processing and Filtering by Geoff Mulligan; Addison-Wesley Publishing Company — A volume that looks at various methods used by email administrators using established tools, such as Sendmail and Procmail, to manage spam problems.*

- *Internet Email Protocols: A Developer's Guideby Kevin Johnson; Addison-Wesley Publishing Company — Provides a very thorough review of major email protocols and the security they provide.*

- *Managing IMAP by Dianna Mullet and Kevin Mullet; O'Reilly & Associates — Details the steps required to configure an IMAP server.*